BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!!
Regex: Closure Properties
Def We will say regex when we mean a lang generated by a regex. For example we might say:

regex is closed under Guido

if Guido was some operation on sets.
Terminology: Regular Languages

**Def** We will say *regex* when we mean a lang generated by a regex. For example we might say:

*regex is closed under Guido*

if Guido was some operation on sets.

We prove closure properties (or say NO, not going to prove it) of regex.
**Def** We will say *regex* when we mean a lang generated by a regex. For example we might say:

*regex is closed under Guido*

if Guido was some operation on sets.

We prove closure properties (or say NO, not going to prove it) of regex.

We already know all of these closure properties since we did closure proofs with DFA’s and NFA’s; however, we are curious which ones can be proven easily with regex’s.
Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

Given regex $\alpha$ of length $n$.

Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert $N$ to a DFA $M$ such that $L(N) = L(M)$. $\sim 2^n$ states.

Swap the final and nonfinal states of $M$ to get $M'$.$\sim 2^n$ states.

Convert $M'$ to regex $\alpha$.$\sim 2^{2n}$ states.
How do you complement a regular language (not a joke)?
While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$. 

Here is how you can do it:

Given regex $\alpha$ of length $n$.
Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.
Convert $N$ to a DFA $M$ such that $L(N) = L(M)$. $\sim 2^n$ states.
Swap the final and nonfinal states of $M$ to get $M'$. $\sim 2^n$ states.
Convert $M'$ to regex $\alpha$. $\sim 2^{2^n}$ states.

Are there $\alpha$ where you get $\sim 2^{2^n}$ blowup? I think so but the literature is unclear on this point.
How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it:
How do you complement a regular language (not a joke)?
While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = L(\overline{\alpha})$.
Here is how you can do it: Given regex $\alpha$ of length $n$.
How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex $\alpha$ of length $n$.

Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.
Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex $\alpha$ of length $n$.

Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert $N$ to a DFA $M$ such that $L(N) = L(M)$. $\sim 2^n$ states.

Are there $\alpha$ where you get $\sim 2^{2^n}$ blowup?

I think so but the literature is unclear on this point.
How do you complement a regular language (not a joke)?
While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex $\alpha$ of length $n$.
Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.
Convert $N$ to a DFA $M$ such that $L(N) = L(M)$. $\sim 2^n$ states.
Swap the final and nonfinal states of $M$ to get $M'$. $\sim 2^n$ states.
How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex $\alpha$ of length $n$.

Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert $N$ to a DFA $M$ such that $L(N) = L(M)$. $\sim 2^n$ states.

Swap the final and nonfinal states of $M$ to get $M'$. $\sim 2^n$ states.

Convert $M'$ to regex $\alpha$. $\sim 2^{2^n}$ states.
How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex \( \alpha \) to regex \( \beta \) such that \( L(\beta) = \overline{L(\alpha)} \).

Here is how you can do it: Given regex \( \alpha \) of length \( n \).
Create NFA \( N \) such that \( L(N) = L(\alpha) \). \( \sim n \) states.
Convert \( N \) to a DFA \( M \) such that \( L(N) = L(M) \). \( \sim 2^n \) states.
Swap the final and nonfinal states of \( M \) to get \( M' \). \( \sim 2^n \) states.
Convert \( M' \) to regex \( \alpha \). \( \sim 2^{2^n} \) states.
Are there \( \alpha \) where you get \( \sim 2^{2^n} \) blowup?
How do you complement a regular language (not a joke)?
While not a joke, there is no easy way to go from regex $\alpha$ to regex $\beta$ such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex $\alpha$ of length $n$.
Create NFA $N$ such that $L(N) = L(\alpha)$. $\sim n$ states.
Convert $N$ to a DFA $M$ such that $L(N) = L(M)$. $\sim 2^n$ states.
Swap the final and nonfinal states of $M$ to get $M'$. $\sim 2^n$ states.
Convert $M'$ to regex $\alpha$. $\sim 2^{2^n}$ states.
Are there $\alpha$ where you get $\sim 2^{2^n}$ blowup? I think so but the literature is unclear on this point.
Regular Lang Closed Under Union

**Easy** The regex for $L(\alpha) \cup L(\beta)$ is $\alpha \cup \beta$. 
Hard Need to convert to NFA’s and do it there and convert back.
Regular Lang Closed Under Intersection

**Hard** Need to convert to NFA’s and do it there and convert back. Might be on a HW or Exam.
Easy The regex for $L(\alpha) \cdot L(\beta)$ is $\alpha \cdot \beta$. 
Regular Lang Closed Under $\ast$?

**Easy** The regex for $L(\alpha)^\ast$ is $\alpha^\ast$. 
Summary of Closure Properties and Proofs

X means **Can’t Prove Easily**

\( n_1 + n_2 \) (and similar) is number of states in new machine if \( L_i \) reg via \( n_i \)-state machine.

\( L_1 + L_2 \) (and similar) is length of regex of \( L_i \) length of \( \alpha_i \).

<table>
<thead>
<tr>
<th>Closure Property</th>
<th>DFA</th>
<th>NFA</th>
<th>Regex</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1 \cup L_2 )</td>
<td>( n_1 n_2 )</td>
<td>( n_1 + n_2 )</td>
<td>( L_1 + L_2 )</td>
</tr>
<tr>
<td>( L_1 \cap L_2 )</td>
<td>( n_1 n_2 )</td>
<td>( n_1 n_2 )</td>
<td>X</td>
</tr>
<tr>
<td>( L_1 \cdot L_2 )</td>
<td>X</td>
<td>( n_1 + n_2 + 1 )</td>
<td>( L_1 + L_2 )</td>
</tr>
<tr>
<td>( \overline{L} )</td>
<td>n</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>( \overline{L}^* )</td>
<td>X</td>
<td>( n + 1 )</td>
<td>( L + 1 )</td>
</tr>
</tbody>
</table>
BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!